Consensus Based Networking of Distributed Virtual Environments

Sebastian Friston[1], Elias Griffith[2], David Swapp[1], Simon Julier[1], Caleb Irondi[2], Fred Jjunju[2], Ryan Ward[2], Alan Marshall[2] and Anthony Steed[1]

Abstract—Distributed Virtual Environments (DVEs) are challenging to create as the goals of consistency and responsiveness become contradictory under increasing latency. DVEs have been considered as both distributed transactional databases and force-reflection systems. Both are good approaches, but they do have drawbacks. Transactional systems do not support Level 3 (L3) collaboration: manipulating the same degree-of-freedom at the same time. Force-reflection requires a client-server architecture and stabilisation techniques. With Consensus Based Networking (CBN), we suggest DVEs be considered as a distributed data-fusion problem. Many simulations run in parallel and exchange their states, with remote states integrated with continous authority. Over time the exchanges average out local differences, performing a distribued-average of a consistent, shared state. CBN aims to build simulations that are highly responsive, but consistent enough for use cases such as the piano-movers problem. CBN's support for heterogeneous nodes can transparently couple different input methods, avoid the requirement of determinism, and provide more options for personal control over the shared experience. Our work is early, however we demonstrate many successes, including L3 collaboration in room-scale VR, 1000's of interacting objects, complex configurations such as stacking, and transparent coupling of haptic devices. These have been shown before, but each with a different technique; CBN supports them all within a single, unified system.

Index Terms—C.2.4.b Distributed applications, I.6.8.e Distributed Simulations, H.5.1.b Artificial, augmented, and virtual realities

1 Introduction

Enabling users to share a virtual world involves communicating that world's responses quickly enough that all users maintain a consistent perception of it; consistent with each other, and with their expectations [78]. The potential for a quick enough response is limited by how fast the technology used to build Distributed Virtual Environments (DVEs) can communicate. Bottlenecks in communication are exacerbated as DVEs scale to larger numbers of users, and larger geographic areas.

The requirement of consistency makes DVEs a problem of scene synchronisation. The first DVEs were designed to facilitate simulations exceeding the limits of one machine, e.g. for cluster rendering. Approaches would distribute authority over scene graph branches, and the system would behave like a distributed database [85]. As computers increased in power, and responsiveness became more important, more of the scene graph was duplicated locally, and the more fine-grained and flexible updates became. There is still a common assumption however of roaming, but absolute, authority. This is embodied in 'ownership' being a principle of many synchronisation schemes – both in original libraries such as *blue-c* [51] and modern ones such as Photon [20]. In Grimstead et al's review, all multi-player games, and all DVEs for which access control was identifiable, used per-object access control [26].

More immersive spaces that better match the real world should allow closer engagement between users. Broll [9] first distinguished between cooperative and collaborative DVEs by their support for concurrent manipulation. Margery et al [49] later classifed cooperation into three levels. Level 1 systems support perceiving other users' existence. Level 2 systems support users modifying a shared world, though not at the same time. The highest level of engagement is Level 3: multiple users manipulating the same degree of freedom, at the same time. L3 allows users to interact as they would in the real world, furthering the Virtual Reality (VR) goal of creating the illusion of interaction without abstraction. L3 is important for believeability, and further, necessary for certain tasks such as the piano movers problem. The piano movers problem refers to the collaborative manipulation of objects too unwieldly for an individual, as might be found in, for example, training scenarios.

Supporting L3 however is a challenge for existing methods of realising DVEs. Synchronisation using discrete updates with absolute authority (transactional) cannot support L3 collaboration. The counterpart to transactional synchronisation is force-reflection, which has been studied for just as long in the field of teleoperation. Force-reflection is a mature technique that involves users exchanging asymmetric variables with a server that integrates their actions and computes a response for all of them. This assumes a client-server model however [36], and the feedback loops created must employ increasingly complex passivity schemes to maintain stability under latency [33].

In this paper, we propose that instead of distributed databases, or teleoperation systems, DVEs could be considered as a distributed data-fusion problem. Based on the observations of plausability and stability that underlie techniques like Position Based Dynamics (PBD) [50], we present a system that brings together prediction, distributed averaging, continuous authority and constraint duplication to synchronise heterogenous DVEs.

In Consensus Based Networking (CBN), simulations exchange state updates along with a metric that approximates the state's information relative to other nodes in the system. Simulations integrate these updates using local solvers, to generate a distributed-

- [1]*Department of Computer Science, University College London.*
- [2]*Department of Electrical Engineering and Electronics, University of Liverpool.*

Manuscript received April 19, 2005; revised August 26, 2015.

average consensus of the state based on some simple criteria such as least-squared error. Our scheme is straightforward and decoupled from any particular simulation technology. This makes it simpler to implement than the distributed mutexes in transactional approaches, and the complex passivity schemes in force-reflection. Beyond the efficacy of L3 collaboration, the nature of the distributed data-fusion problem presents many options for managing large-scale simulations. The inherent support for heterogeneity could be an advantage in administering such simulations and increasing their scope, and the focus on local autonomy could give users more control over their personal experience of the shared space.

Our motivation is to enable L3 interaction in large virtual worlds, and to do so in an accessible and easy way. Part of accessibility is generalising to different ways of building worlds & applications, and different modalities. We therefore aim to address three problems in this paper: supporting L3 interaction, scalability of shared simulations, and the asymmetric integration of haptics.

We implement our method in Unity and demonstrate its practicality through a number of applications that show it addressing common problems in DVEs – including causality preservation, interactions between large numbers of shared objects, complex arrangments such as stacking, L3 collaboration, and support for haptic feedback – simultaneously in the same system under various network conditions.

2 Previous Works

2.1 Transactional Scene Synchronisation

Much of the early work on scene synchronisation focused on network architectures for delivering updates and APIs for seamless integration (e.g. [25] [34] [47] [51] [11] [92] [41] [74] [83]). The first problems addressed were how to build simulations that could scale beyond the processing power of a single computer, avoid saturating network connections, and leverage existing scene graph libraries. For example, for the purposes of cluster rendering [85]. To manage consistency as simulations became parallel, schemes use the concept of authority that defines which part of the graph is owned by which process. Processes have complete control of their branch and send unilateral updates.

In peer-to-peer systems, nodes synchronise with approximations of traditional synchronisation primitives such as mutexes [19] or message-pumps [65]. Other techniques approximate collaborative multi-tasking [10] [51] or simply nominate an authority for the whole graph, task permitting [59]. Jorissen et al [39] proposed a system in which every object was autonomous, and all interactions, such as collisions, manifested as events. These schemes approximate distributed transactional databases.

Fujimoto [22] provides a detailed overview of issues in Parallel Discrete Event Simulation (PDES), and the extension of loosely synchronised PDESs to interactive simulations. As updates are absolute and atomic, purely transactional schemes cannot support L3 collaboration. Coherency depends on network quality. The more outdated or infrequent updates become the larger the divergence between branches on different nodes. This can manifest in a number of ways. The system can appear unresponsive, corrections become larger and more noticable, and inconsistencies can appear in the order of causality. A number of techniques have been developed to reduce inconsistencies, or otherwise reduce the symptoms, including synchronisation filters, local perception filters and rollback.

2.1.1 Synchronisation Filters

Synchronisation filters pre-process or otherwise change now discrete updates affect the local state. One use of synchronisation filters is to maintain causality. Roberts & Sharkey [62] introduced the concept of *sufficient-causal-ordering*, in which the application dictates dependencies between events. Events can be re-ordered before being integrated to maintain causality. Choukair et al [14] presented a system that used meta-data to re-order updates based on importance, rather than time. Allard et al [1] filtered and re-sampled messages to couple nodes with heterogenous types and speeds. Latoschik et al [43] used messages to generate secondary signals and events.

Synchronisation filters can also be used to support L3 collaboration. For example, Wolff [88] averaged object transforms between concurrent inputs. Ruddle et al [66] experimented with two ways of integrating concurrent commands: taking the common component or taking the mean, and found which was superior was not only task-dependent but task-stage-dependent. Anthes et al [3] proposed a dataflow architecture in which continous parameters like positions were fed through a distributed chain of filters, including filters that could merge the outputs of other chains.

Synchronisation filters are a very general concept. Our implementation could be considered a type of Synchronisation Filter. However, pre-processing depends on the semantic of the variables being exchanged. For example, force could be interpolated, whereas player health could not. Anthes et al [3] suggested the messaging system be aware of this distinction, and support different mechanisms for continous and discrete updates.

2.1.2 Local Perception Filters

Sharkey et al [76] first introduced *local perception filters* as a mechnaism to dilate time by controlling object speed. From the perspective of each user, objects would move faster in their local vicinity, and slower as they approached a remote user. By slowing objects down, the technique limits how far they can deviate due to a remote user's action, before updates from that user are received.

Ryan & Sharkey [67] further defined causal surfaces, and so critical speed. Critical speed relates the distance between objects in time to message latency. Effectively, the speed at which two objects can approach, above which they will meet before messages between them can arrive, resulting in potential discontinuities in causality.

2.1.3 Rollback

While effective [68], local dilation can conflict with other goals of an application, especially highly responsive ones such as first person shooter games. Local replication can be extended to include client-side prediction of actions over which local simulations do not have authority.

For example, Cronin et al [16] maintained multiple optimistic simulations, running at various delays. In the case of delayed updates, the state could be 'rolled back' to one of these other simulations in order to integrate the update. Bernier [6] described how rollback was used for latency compensation in Half-Life. Clients would perform local simulations, while also sending their inputs to an authoritative server. On receiving an input, the server would roll back using a short history, to integrate the input into the state at the time it was generated.

When rollback does take place, or an authoritative update is pushed, artefacts can be seen by users. In many applications these

are not noticable (e.g. [60]), however for larger corrections different techniques may be needed. For example, Savery & Graham [73] tested different corrections to position discrepancies in games, and found it was best to minimise velocity. Singhal & Cheriton [77] augmented a predictor with an adaptive convergence algorithm based on path interception.

2.2 Uncertainty

Some approaches consider uncertainty as a property in and of itself. For example, Coelho et al [15] stored covariance matrices (each transform being the mean of a probability distribution function, rather than the true transform) with the intent that these be reflected in the behaviour of the application. Ryan & Sharkey [69] used local distortions to show the state propagation over their previously defined causal surfaces.

2.3 Input Mirroring

Roberts & Wolff [63] proposed extending local replication to include behaviour models. Input mirroring (or Active Replication [46]) involves duplicating a simulation, and broadcasting all inputs to all instances, so that over time all nodes receive the same inputs. For this to work however, the simulations must be deterministic, and the inputs processed identically at each node. This is difficult with traditional physics simulations which are highly time-step dependent. All example systems leverage some special case of the application to make this tractable. For example, Gunn et al [28] [27] created a shared surgical simulation, but the properties of the tissue were such that second order motion parameters could be ignored, minimising divergence. Riot Games' League of Legends [37] and other multiplayer games [81] use a deterministic distributed simulation with global time, but only a subset of the state that covers interactions between players and the game is simulated like this (*gameplay determinism*).

2.4 Prediction

Delay is inherent in current technology. Task or type specific integrators (e.g. [66]) can ameliorate effects of delay. For consistency though, the only viable solution once an optimal architecture has been found, is predictive compensation [40].

2.4.1 Extrapolation

The simplest predictors are Euler integrators of various order (e.g. [58]). Gutwin et al [29] evaluated linear prediction in telecursor motion. They compared hypothetical predictors with errors up to those of a reference linear implementation in a user study, and found little difference in user performance until 25-50% of the error was reintroduced. Brandi & Steinbach [8] used linear regression to compute the gradient for predictors to compensate for packet loss and potentially noisy haptic inputs. Zaeh et al [91] used a Double Exponential Smoothing Predictor for update reduction.

Singhal & Cheriton [77] were one of the first to use predictors for delay compensation in DVEs. They extrapolated motion based on previous time-stamped samples, including a second-order predictor to support curved motion and a convergence algorithm to reduce spatial jitter. Hanawa & Yonekura [30] analysed zero, one, and second order interpolation polynomials in Lagrange form[1]. Hanawa & Yonekura [31] followed up with a Taylor expansion of a variable performed on the server, based on the observation that the server could sample with smaller time-steps than the client.

Tumanov et al [82] utilised an Extended Kalman Filter (for the non-linear rotation parameters). Tumanov et al seemingly used their Extended Kalman Filter (EKF) to compensate for process noise, but such advanced predictors could model knowledge about sample and network delay properties, and even local inputs. Hinterseer et al [35] used a Kalman Filter to extrapolate haptic power variables (force, velocity), but to reduce update counts. The prediction did not compensate for latency; a second linear predictor was used for this. Kalman filters automatically compute the confidence in their own estimates, whereas with traditional predictors this must be done by, e.g. monitoring update delays [7]. For physically-based problems, the predictive part of a Kalman filter is often a typical Euler integration. More advanced filters could integrate out of sequence measurements.

Schuwerk et al [75] adapt the local force rendering algorithm to take into account remote motion constraints, the haptic device velocity, and the network delay. This showed excellent results in a user study, but constraints for only a couple of object configurations were tested. The motion constraints could handle, for example, an object on a surface, but it is not clear what the effect would be on, e.g. stacked objects.

Prediction can also ameliorate latency indirectly. Sandoz et al [70] extended the bounding-sphere distance function to include first-order dynamics, and so predict the time at which objects would collide in order to pre-emptively request ownership and reduce the apparent overhead of the synchronisation.

2.4.2 Proactive & Reactive Prediction Schemes

Tumanov et al [82] differentiate between proactive and reactive prediction. Proactive schemes anticipate network delay, such as by extrapolating a state for its expected time of receipt. Reactive schemes extrapolate previously received updates for the current time. The distinction is fuzzy since predictors used in reactive schemes such as Predictive Contract Agreement Mechanisms (PCAMs) can model network latency.

Cheong et al [38] used Smith Predictors to apply additional forces to haptic objects to keep them in sync with their remote counterparts. The controller of the forces incorporated the round-trip-time of position updates. Schuwerk et al [75] incorporated estimates of delay into their adaptive force rendering algorithm (Hooke's Law) parameter.

The distinction raises an important dichotomy between extrapolation that accounts for network latency, and that which accounts for message frequency. This is effectively whether an update's epoch is the time of transmission or the time of receipt. Where clock synchronisation is available, the former will intuitively have better accuracy. Clock synchronisation is not always available however, in which case proactively compensating for link delay, or extrapolating across frames between infrequent updates may be the best solution. Not all papers make clear (e.g. in [58] [29] [31]) which method is used.

Prediction is key to CBN because it improves convergence speed. Without prediction simulations will undergo damping proportional to delay. As in other systems, there is the risk of prediction error. Which predictors are best to use in CBN is an important topic for future works. CBN's duplicate constraints do offer some mitigation of the error from simple predictors however. Our prototype is built using linear extrapolation.

1. Effectively, Euler integrators with the motion parameters implicitly defined from the position history.

2.5 Force Reflection

One of the issues with transactional scene synchronisation is that it cannot support collaborative manipulation. This is what Margery et al [49] refer to as *Level 3* – multiple users manipulating the same degree of freedom, at the same time. Commonly, L3 clients send continuous asymmetric inputs to an authority, which integrates them and sends an updated state in an approximation of bilateral teleoperation.

Bilateral teleoperation, also known as force reflection, exchanges position (or its derivatives) and forces between a user and a real or virtual remote world (e.g. [42] [2] [61]). This works well in many scenarios, but the need for a single authority limits the scalability of the scheme. Furthermore, discretisation and sample-hold effects can quickly destabilise naive implementations that directly transmit force [44] [52]. One way to compensate for this is to passivate the channel. A passive system is one that requires more energy to be introduced than can be removed from it. A considerable body of work in teleoperation has been dedicated to this, which has resulted in a number of schemes to support passive force-reflection with minimal damping accross imperfect connections (e.g. [53] [44] [4] [45] [80] [5]).

2.6 Virtual Couplings

Virtual Couplings bind remote instances of objects with controllers such as Proportional-Integral-Derivative (PID) controllers. Virtual couplings are a popular control scheme for bilateral force reflection [71] [64] but could be applied to general synchronisation. Simulations create physically-based (e.g. Hooke's Law) constraints between objects over the network. Objects respond immediately to the local simulation, while authority is implicitly distributed. In a telehaptic system the force and position variables are no longer out of phase locally, but the behaviour of the object itself is now dependent on constraints that are functions of network properties, introducing inertia. The weighting of these constraints is a realisation of the *consistency-responsiveness trade-off* [18].

CBN has much in common with virtual couplings, and it is easy to imagine a constraint-mirroring implementation of virtual couplings. The difference is that virtual couplings enforce coherency through the physics simulation. In CBN the state is simply re-evaluated. Therefore CBN corrections do not introduce energy into the simulation, nor do they inherently suffer from distributed stiffness.

2.7 Summary

Systems can be characterised by which variables they transmit, and how they handle authority. These loosely coupled characteristics determine the level of cooperation a system can support: whether - and where - updates can be integrated, or whether they are mutually exclusive. Grimstead et al's review [26] of collaborative systems circa 2005 provides an overview of how many systems separated into these categories and reflected a definite preference for client-server architectures.

DVEs have often been considered as distributed databases. For L2 a number of scalable architectures have been introduced. However the synchronisation mechanisms that maintain consistency prohibit L3 collaboration and can reduce responsiveness under latency. DVEs have also been considered as large-scale physical systems. This enables L3, but introduces the same stability issues faced by teleoperation systems, and the client-server architecture limits scalability.

We propose that the problem of state-synchronisation is actually one of distributed data-fusion. In the next section we show how position-based approaches to prediction, consensus, constraint duplication, and continous authority can combine to build highly immersive and responsive worlds.

3 Consensus-Based Networking

CBN is inspired by Virtual Couplings, Distributed Consensus and PBD. PBD is a physics simulation approach that solves constraints at the position level, rather than integrate forces. The advantage is that simulations are inherently stable. In CBN, autonomous simulations at each node run in parallel and constantly exchange their states. Nodes integrate others' states into their simulation at the position level in a continuous manner, averaging out differences and pulling all simulations towards a consensus over time.

Figure 1 illustrates the logical implementation for node j on a network of n nodes. Each node integrates its peers' states s_{ni} using weighted averaging in a local solver (consensus solver). The physics simulation (physics solver) runs against these averaged states. The use of weighted averaging means authority can be continuous. The authority of an update depends on its relative information with respect to the local simulation. Divergence between updates is limited by duplicating as many constraints (C) as possible between simulations, so that the physics solvers follow similar trajectories. Therefore the only updates with high information should come from truly stochastic inputs, such as the users. The use of continous authority however also supports a constant exchange of low information updates, that maintain consistency over time between non-deterministic simulations.

In Figure 1 the physics solver and consensus solver are shown as one stage, as how they interact will depend on the physics simulation technique (see Section 4). Indeed, in some cases they could be combined. The solvers work together to compute the best estimate of an object's state s_t, based on the current estimate s_{t-1}, local constraints C_j, and remote C_{ni} constraints & states s_{ni}. Updates are transmitted and received asynchronously. To compensate for latency and low update rates, both state and constraint updates are timestamped and extrapolated using local prediction (s'_n, C'_{ni}). If this were not done, delayed updates could introduce damping.

Each node transmits it's post-solve estimate, which is a product of its own simulation, and other nodes' estimates. Therefore with each exchange the difference between simulations should reduce, until they converge to the same state and trajectory.

A comparison of CBN with other approaches is shown in Table 1. In this section we introduce five concepts that underlie CBN: (i) Distributed Consensus, (ii) Input Weighting, (iii) Prediction, (iv) Constraints and (v) Hard and Soft Inputs.

3.1 Distributed Consensus

Synchronising physically-based simulated objects is very similar to the problem of consensus in multi-agent systems [56]. Consensus algorithms are interesting because they will provably converge even with non-ideal communications [90]. Garin & Schenato [23] provide a good introduction to distributed consensus. A number of algorithms for physically based bodies [24] [72], including with non-linear constraints [55], have been demonstrated. In addition, constraint-based approaches for non-physically based optimisation problems have been addressed [48].

Distributed consensus typically means some form of distributed average. Average-consensus could be interpreted as state-average

Scheme	L1	L2	L3	Client-Server	Peer-to-Peer	Loose Synchronisation	Asymmetrical Inputs	Symmetrical Inputs	Arbitrary Constraints	Non Physical Solve
Transactional	✓	✓		✓	✓		✓			
Force Reflection	✓	✓	✓	✓		✓	✓			
Virtual Couplings	✓	✓	✓	✓	✓	✓		✓		
Consensus Based Networking	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABLE 1
Feature comparison of Consensus Based Networking with other popular schemes.

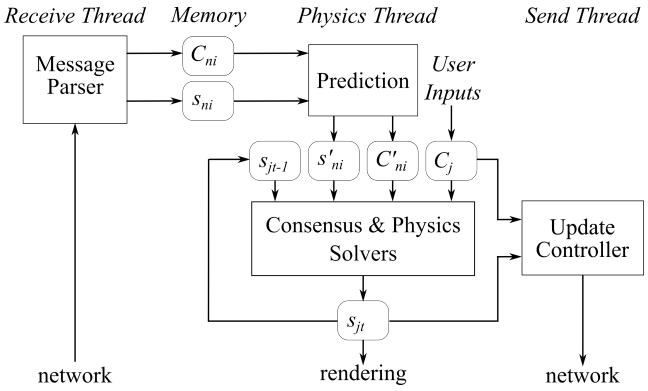

Fig. 1. System Overview for a Single Node

in DVEs. Such averages have also been expanded for time-delays [57], and it has been demonstrated how weight control can change the behaviour of distributed systems [89].

While consensus algorithms have a lot of overlap, they differ from the problem of DVEs in some key ways. For example, much work is on graph connectivity, which we can decouple in DVEs. Further, while consensus algorithms will converge, the problems they address often don't have the interactivity requirements of how *fast* they will converge.

3.2 Weights & Influence

If two simulations evolved identically, they could maintain consistency without ever communicating; they only diverge when one receives an unpredictable input. This is the principle behind PCAMs [17]. The more advanced the predictors employed, the less consequential latency is. The weighting of updates describe how unpredictable they are, and therefore the confidence a simulation should have in overriding its own estimate. For example, if a user interacted with a simulation in a way that was not mirrored at any other node, only that simulation would be aware of the action, and so its updates would have higher authority to override remote simulations - that simulation has more information about the state of the global system.

In some DVEs users really were the only stochastic inputs, such as the peer-to-peer lockstep simulation in Doom [78]. Deterministic simulations are difficult to build for general purposes however. Instead, CBN sends a continous stream of updates to maintain consistency under non-determinism. These updates however will have noise themselves, from the same sources of non-determinism and from prediction error. Naively weighting all updates equivalently would introduce spatial jitter and possibly damping. The updates are therefore weighted much lower to allow the local simultion to remain responsive and stable, corrections implicitly being applied over time.

The actual information, noise or error budget of a process can be quantified exactly. In this early work we do not do this, and so call our weightings a measure of *influence*.

In our prototype we hand-pick baseline influences that work well. Simple policies can adjust these: for example, if the user applies an asymmetric constraint with a controller, the constrained object's influence should increase. We hope future work can compute influence automatically on a proper information-theoretic basis.

3.3 Prediction

Expressing a weighted average as an interpolation, it is easy to see that if the prediction were perfect, the weight would be inconsequential. Conversely, performing weighted averaging using an outdated state effectively introduces error equivalent to that due to the latency.

CBN updates include additional-order terms that allow local simulations to extrapolate a state or constraint to the simulations local time, compensating for latency. While both states and constraints can be extrapolated, their effects on the simulation are different: outdated states result in damping, while outdated constraints result in active divergence.

Prediction is highly significant to CBN. In some ways it is orthogonal in that CBN doesn't rely on prediction to function, but damping is dependent on predictor quality. As prediction error will change with time, influence is also a function of time for each update, and should decrease with increasing prediction intervals.

Our implementation uses first-order Euler's method. Another effective but cheap predictor would be a Taylor Series expansion, with a frequency limiting check to bound error. An advantage of CBN is that by adjusting the influence, predictor accuracy can be traded off for consistency without affecting responsiveness. Alternatively, limiting extrapolation time can trade off responsiveness for stability.

3.4 Constraints

Constraints refer to anything that controls the evolution of the simulation non-stochastically. Straightforward examples would be the actual constraints in PBD, colliders, motors & joints, collision response, and numerical integration settings. Any differences between these will be a source of inconsistency (noise or bias).

The more constraint duplication there is, the simpler the weighting scheme and predictors can be. If user manipulation of an object causes a chain reaction, only the first object, or the manipulation itself, needs to be transmitted with high influence. This is because the same reaction will be initated remotely due to reproduction of the initial object movement. Constraints can be used as a mechanism for user input. For example, by creating a spring between a controller and an object as in Section 4. In this case the inherent constraint mirroring is sufficient to mirror the input and its effects, and the object weightings are unchanged.

3.5 Inputs

Changes to the shared consensus can be made through changing the state of shared objects directly, or the shared constraints on them. There are then two approaches to user input. In the first an object is manipulated by a mechanism known only to one simulation. That simulation pushes an update with higher influence to force the consensus closer to its own state. In the other, a node creates a mirrored constraint. When implemented by remote simulations, their states will converge as a result of the constraint. What controls the constraint is immaterial, so long as the constraint itself can be duplicated. We refer to these approaches as *hard* and *soft*, respectively.

This allows heterogenous simulations to cooperate, with different levels of abstraction between the Virtual Environment (VE) and the shared simulation. For example, CBN could transparently connect a user with a VR controller to a user with a keyboard and mouse, to a user with a haptic device. Similarly, the haptic device's loop can run entirely locally, maintaining stability and improving stiffness, as in virtual couplings.

The two cases are not equivalent however. If two nodes have asymmetric constraints as in the first case, the coupled simulations effectively become one large numerical solver, as compared to two synchronised but distinct simulations in the second. This places far higher requirements on the network. It is always better to duplicate constraints if possible. We demonstrate both cases in Section 5.

4 Implementation

To demonstrate the real world efficacy of CBN we constructed a number of shared environments in Unity 2019.2. How the consensus and physics solvers interact depends on the physics simulation technique. For example, if PBD were used, the remote states s_n could be represented as remote position constraints; other constraints would have their native representation, and both local and remote constraints would be solved together at the same time. To show the practicality of CBN however our prototype uses PhysX, a point-mass impulse-based engine that is in-built into Unity and used to provide physics for many VEs.

As PhysX is an impulse-based engine, solving is a two-step process in which we update the state (position and velocity) using weighted-averaging (consensus solve), then perform a simulation step in PhysX (physics solve). That is, state updates are handled like position constraints by the consensus solver, and other constraints are handled by PhysX - as joints and colliders that are dynamically duplicated at each node. The process-flow is shown in Figure 2.

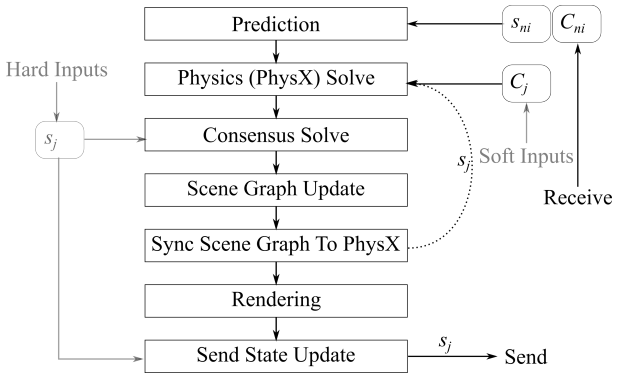

Fig. 2. Implementation Process Flow

4.1 Updates

Nodes communicate using TCP. Message passing is facilitated through the Scene-Graph-as-a-Bus technique [92]. Each instance of a shared object has a consistent Globally Unique Identifier (GUID). In our prototype this is set at design time. Messages are passed through the root-node to a Router object that performs fan-in and fan-out of messages from other nodes connected in a peer-to-peer fashion. The scene-graph itself is responsible for routing update messages to their correct node, where local components evaluate and integrate the update. The network architecture is completely decoupled from CBN. It is assumed that any object having a particular GUID receives all updates from other objects with that GUID, across all nodes.

Type	Property	Notation
int	SharedObjectGUID	G
Vector3	Position	x
Vector3	Velocity	v
Quaternion	Orientation	q
Vector3	AngularVelocity	$\dot{q}$
float	Influence	w
long	Timestamp	t

TABLE 2
Update message members

The CBN state update (s_{ni}) message is shown in Table 2. Updates are stateless. Nodes routinely transmit their own estimates according to a policy. In our prototype nodes transmit at a fixed rate. Other message types are defined for shared constraints. Currently our system only defines one: a spring joint. This message simply communicates the constraint parameters.

4.2 Consensus Solver

Updates are represented as state-constraints in a PBD-style per-object solver. On each timestep updates are re-evaluated to compute a predicted state and influence. In PBD the energy gradients of arbitrary constraints are evaluated with respect to the state variables, so a new state can be computed that minimises the energy. For state-constraints, distances in state-space can be directly compared. Since we use a separate physics engine, state updates are averaged in our solver, while other constraints will be either colliders or joints that are mirrored in the physics engine.

The local estimate is represented as a state-constraint (C_{sj}) with fixed influence. It's parameters are read from the physics engine before the solve. A linear weighted average is performed on the extrapolated states using influence for the position and velocity components (Eqn. 1). For clarity, the state and node subscripts s, i have been dropped for Equations 1, 2 & 3.

$$x = \frac{\sum C'_x C'_w}{\sum C'_w}, \quad v = \frac{\sum C'_v C'_w}{\sum C'_w}, \quad \dot{q} = \frac{\sum C'_{\dot{q}} C'_w}{\sum C'_w} \tag{1}$$

$$q = ||E_{\max\lambda}\left(\sum (C_q \otimes C_q) C_w\right)|| \tag{2}$$

Orientation averaging is more challenging as it is non-linear and the weights do not apply in the straightforward way they do for positions. Any implementations must also be fast in order to support many shared objects. Hartley et al [32] and Chatterjee & Govindu [12] provide good introductions to rotation averaging. Our implementation uses the Eigen Vector Decomposition (EVD) method (Eqn. 2). State-updates are locally extrapolated using a simple Euler extrapolation (Eqn. 3). Angular velocity ($\dot{q}$) is

averaged (Eqn. 1). It is not used to extrapolate the orientation C_q, but is applied to PhysX in order to reduce divergance during the PhysX step.

$$C_x = C_x + C_v \cdot (t - C_t) \tag{3}$$

4.3 Hard and Soft Inputs

Hard inputs directly set both the local estimate and the estimate that is transmitted to the remote nodes: the transmitted state does not depend on the solve and physics is effectively disabled. The rendered state is the direct weighted-average of all hard state-constraints. To do this the state must be overridden at specific times in the process flow, to ensure the correct states are shown to the user and transmitted to remote nodes (Figure 2). Soft inputs use physically-based constraints such as springs to manipulate the consensus through the physics simulation.

Both modes support L3 collaboration. Hard inputs do not have any simulation abilities, but are perfectly stable because nothing is reflected. Soft constraints require the input mechanism to go through the physics engine, potentially providing a better user experience. Which is optimal will depend entirely on the application.

4.4 Time Synchronisation

As predictions are functions of time, it is important that simulation time be consistent across all nodes. Newer operating systems support APIs that give system time with microsecond precision. The absolute wall clock time does not matter so long as it is consistent, however the farther the distance between the nodes the farther away the reference has to be to cover all of them. Synchronising system clocks was done with Precision Time Protocol (PTP) [87]. This can deliver time with nanosecond precision with the correct physical layer implementation. Both sites in our implementation synchronised to UTC from GPS Grandmaster clocks.

4.5 Physics Time

The clock that performs the state extrapolation (Eqn. 3) must also be in sync with that of the physics engine, as any differences will manifest as prediction error during the solve (Eqn. 1). The interval $C_t - t$ (Eqn. 3) is referenced to wall-clock time, so the physics engine must run as close to wall-clock time as possible.

By default, Unity polls a timer each frame to see if an interval has been exceeded, and if so steps PhysX with a fixed timestep independent of wall-clock time. This causes the effective physics simulation time to drift. We disable this behaviour and step PhysX ourselves in order to match the physics time to system time. Our implementation tracks physics time as a sum of all previous timesteps. In PhysX, only timesteps within a limited interval (8 ms to 32 ms) can be used for numerical stability reasons. At the start of each frame, we step in increments within this range as many times as possible before the physics time exceeds the system time. We make this time available to the rest of the implementation, for example to make predictions or timestamp messages. In practice, it is more important that the clock does not drift, than that individual timestamps are consistent to the millisecond.

4.6 PhysX Integration

Once the state has been solved, it is applied back to the PhysX simulation. PhysX is an impulse (force) based simulation so this must avoid introducing implausible energies. To avoid introducing quantisation noise from solver, updates are only applied if they exceed a threshold approximately that of PhysX's sleep threshold (0.001 m for position, $0.01\,\mathrm{m\,s^{-1}}$ for velocity and 1° for rotation).

We use the rigid-body *MovePosition* & *MoveRotation* methods, which update the position without recalculating the velocity. An issue with these is that for non-kinematic bodies, continous collision detection is not performed for the transformations. We were unable to find a workaround. For the most part this did not affect our system, because continous collision detection is still performed for the PhysX simulation step (the 'soft' inputs). However artefacts became noticable during the high-speed *Tennis* scenario under high latencies.

5 Validation

To demonstrate the practicality of CBN, we implemented a number of collaborative virtual environments that demonstrate how CBN adapts to common challenges in DVEs.

5.1 Sphere Cliff

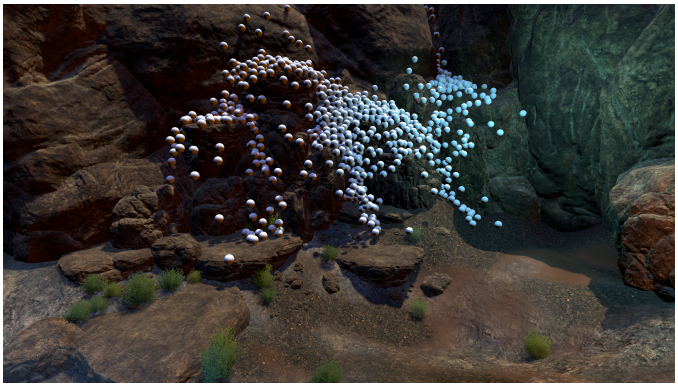

Fig. 3. Cliff scenario after running for a few seconds.

In this scenario 1000 spheres are dropped off a virtual cliff with complex collision geometry (Figure 3). This demonstrates the ability to maintain consistency with large numbers of objects. As the balls can collide, a transactional system would either have to be client-server based, or constantly exchange authority, massively slowing the simulation as ownership was transferred. Even L3 systems that dynamically exchanged constraints or events pairwise would be challenged. In our implementation, the only objects exchanging messages are the balls, and each only has to exchange messages with its counterpart. This was the technically simplest scenario, with no user input. The CBN implementation does not break down or introduce artefacts despite the large number of collisions, regardless of network profile.

5.2 Room

In this scenario two users interact in a prototypical real-world scale VE (Figure 4). This demonstrates L3 collaboration in close proximity. Users could see each others' actions, hand over objects, collaboratively pick up objects, and see the consequences of secondary effects. Traditionally this would be implemented

Fig. 4. Room scenario in starting configuration.

with force-reflection. The dynamics could change with network quality as these directly affect the feedback loop, which may need to be adjusted per-profile to maintain stability. In CBN both nodes are identical and interaction is supported through constraint duplicatation. Interaction via the controller was force-based with three loose springs, in order to create an illusion of weight. The springs were duplicated at each node (*soft inputs*, Section 3.5). Simple avatars were implemented by creating shared objects whose influence was skewed towards the representative party. There was no observed instability in any profile.

5.3 Tennis

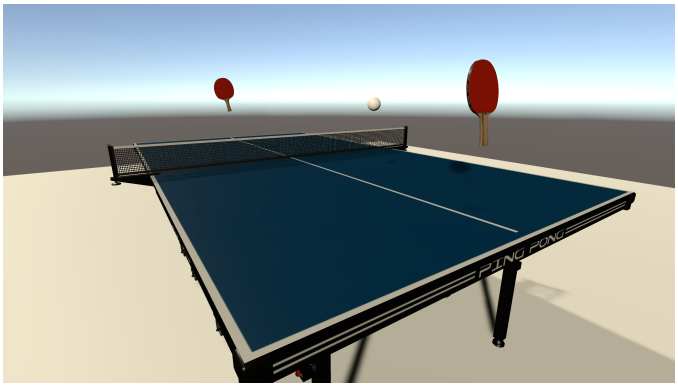

Fig. 5. Tennis scenario in starting configuration.

In this scenario two users played a game of table tennis (Figure 5). This demonstrates how autonomous simulations can help preserve causality. Traditionally, this would be implemented with a transactional system, as the ball does not require concurrent manipulation. However latency could cause discontinuities in causality when it is hit by the remote user. Local perception filters [76] and predictive collisions [70] could ameliorate the effects of latency. In our implementation, each user has full authority over their racket and shared authority over the ball. Since both rackets in any simulation are physical, a collision response can be computed and displayed locally. The ball appears to respond to the remote user's action, even before they send their version of the collision response.

This demonstration worked well in low latency conditions ($< 28.5ms$) but our implementation undermined the goal in higher latency conditions. This was because rackets updated with the *MovePosition* APIs did not support continutous collision detection, resulting in tunnelling. The game maintained stability, but was unplayable above Profile 3 (28.5 ms) as the ball did not move smoothly enough to intercept. In the future this could be helped by reducing the physics timestep and performing interpolation for intervening render frames. The influence would likely need to be reduced as well.

5.4 Piano Movers Problem

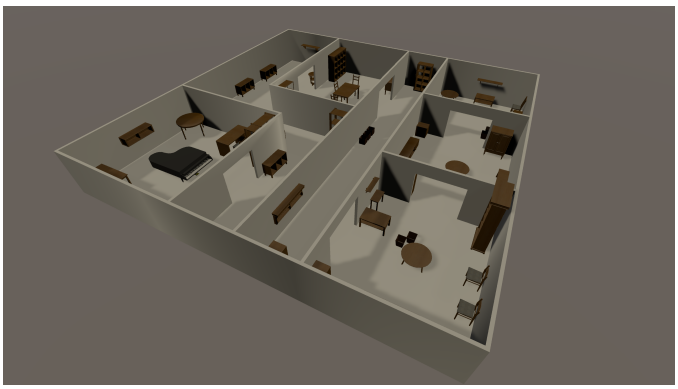

Fig. 6. Piano scenarion in starting configuration. The piano must be moved to the opposite corner of the environment.

In this scenario two users engage in a Piano Movers Problem mini-game using two Phantom Omnis (Figure 6). This demonstrates seamless support for haptic feedback. Haptic feedback is typically modelled with Hooke's Law and supported in teleoperation applications through bilateral force reflection. In our implementation, grasping the piano creates a single spring constraint. That is, an individual Phantom cannot exert torque: users must rely on help from their counterpart to successfully navigate the piano through the maze. This demonstration could be supported by mirroring the spring constraints created by the Phantom's integration, but for the sake of a more interesting demonstration, we chose to directly manipulate the state as described in Section 3.5 (*Hard Inputs*). CBN itself does not have any awareness of the Phantom or how it is integrated.

Even under the highest latency conditions a good experience was enabled. This was likely helped by the low forces and object speed, and the Phantom's mechanical compliance filtering low-level jitter.

5.5 Stacked Cubes

Glenn Fiedler's Stacked Cubes is a state-of-the-art demonstration created for Oculus which involved synchronising two PhysX instances [21]. In this scenario we re-create the demonstration using CBN. The original used roaming authority, where peers took ownership of a subset of cubes and routinely transmitted updates that overrode their remote states. Our implementation shows how even without an authority, CBN simulated objects can remain stable in challenging configurations such as stacking, both statically and during a coordinated lift of the base. Controller interaction was force-based as in the Room scenario (5.2). While visually different this demonsration is technically the same as Room. Consequently it should perform identically and any differences in objective measures would reveal application-specifc depdendencies of our technique.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3052580, IEEE Transactions on Visualization and Computer Graphics

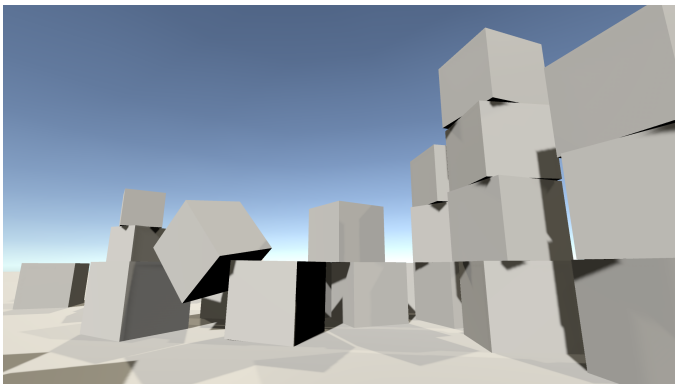

Fig. 7. Cubes scenario after pushing over one stack.

6 Evaluation

Many quantitative architecture analyses focus on frame rate or bandwidth (e.g. [86] [1] [41] [19]). In bilateral force reflection, transparency and stability are the typical measures (e.g. [75] [71] [64]). For synchronisation schemes, most studies evaluate variable consistency, especially when predictors are involved. Inputs are either synthetic (e.g. [58] [77]) or captured user data (e.g. [82] [91]). As a synchronisation scheme, our evaluation focuses on consistency, with respect to network properties and predictor accuracy.

6.1 Profiling

The five scenarios (Section 5) were run over a dedicated fiber link. A network emulator (JAR Emulate Lite) was placed in the link allowing us to control the Quality of Service (QoS). Two experts participated in the interactive environments, ensuring L3 collaboration took place. During these sessions the shared object states at each end were recorded.

6.2 Network Properties

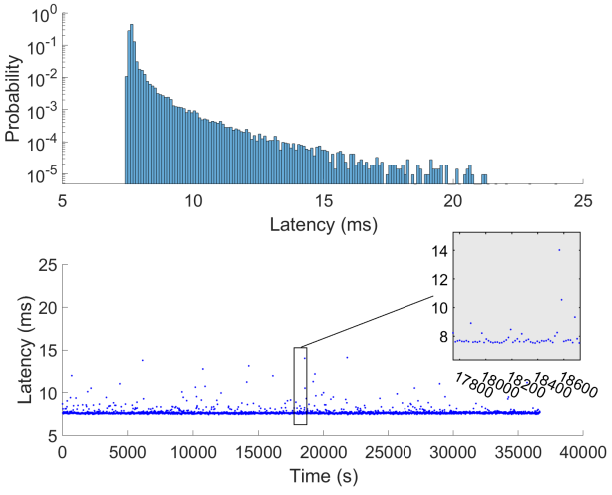

Fig. 8. Latency profile of dedicated fiber link. Logarithmic probability distribution (top) and point-wise measures (bottom).

The dedicated link exhibited the latency distribution shown in Figure 8. For each scenario, five sessions occured with different network conditions. The network emulator degraded the connection according to four profiles approximating DVEs of different scale (Table 3). Profiles were defined based on worst-case ping statistics between real-world cities[2]. Jitter was highly exaggerated to represent a worst-case scenario [84] [54] [13]. For these tests we did not emulate packet-loss, as we used TCP and so packet-loss would manifest as jitter. Anecdotally, severe jitter does result in severe artefacts. In the future moving to UDP should be investigated as our scheme should be inherently tolerant of packet loss.

Profile	Additional Delay (ms)	Latency (ms)	Ping (ms)	Jitter (ms) One Way	Jitter (ms) Round Trip
1 (Dedicated)	0	3.5	7	0	0
2 (City)	2.5	6	12	0.5	1
3 (Country)	25	28.5	57	2.5	5
4 (Continental)	60	63.5	127	4	8
5 (Intercontinental)	120	123.5	247	5	10

TABLE 3
Network Profiles

6.3 Influence

Remote influence was set at $5-10\%$ for all shared objects. For single-user objects such as avatars or the grasped tennis rackets it was set to 100% by explicitly coded policies.

6.4 Results

6.4.1 Consistency

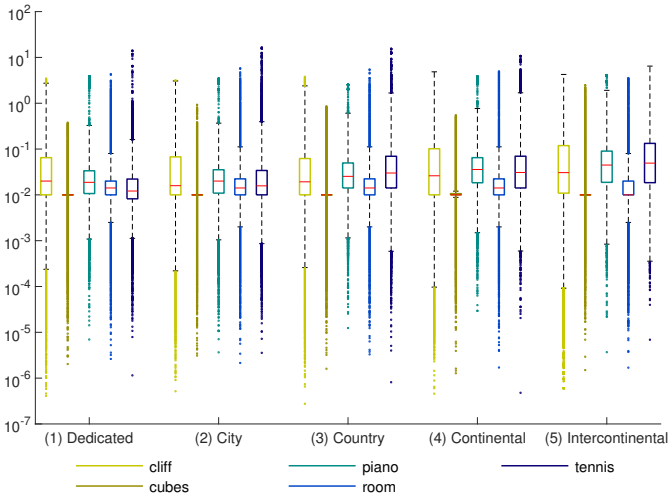

Fig. 9. Position divergence in meters between two nodes for each session (Logarithmic).

Figure 9 shows the position incoherencies for all sessions. This is the Euclidean distance between the x, y, z coordinates of an object's instances at each node. Each plot includes all shared objects over all frames. Outliers were considered 1.5x the interquartile range. Position divergence is both profile and application dependent. Excluding outliers, consistency is typically between 1 cm to 10 cm. The Piano Mover scenario has the highest variance across all profiles, which is not surprising because it is the only one with

2. https://wondernetwork.com

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3052580, IEEE Transactions on Visualization and Computer Graphics

asymmetric constraints. The Tennis scenario steadily degrades with network quality. This is also unsurprising as it has the highest object velocities and the largest number of fast, stochastic inputs. Where avatars are supported, we can expect a number of outliers. This is because the avatar components count as shared objects, but locomotion is enabled through teleporting, which obviously cannot be predicted. For example, Figure 10 shows the divergences for every object (across two plots for clarity) in a scenario for the duration of one capture (Room, Profile 2). The first six outliers (blue labels) belong to the avatar. The second set of outliers (red/orange labels) are objects that penetrated the floor (Section 7.7) and fell in sync, but at increasingly high speeds. All scenarios maintained stability regardless of network quality. As expected, Room and Cubes had very similar performance.

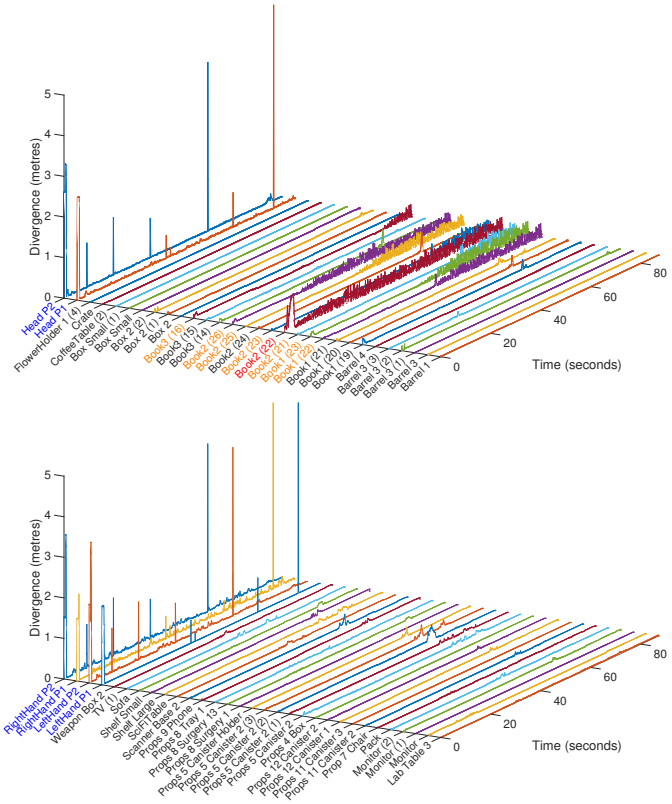

Fig. 10. Position divergence between two nodes for the Room scenario in Profile 2. Objects have been split across two plots for clarity. Red/Orange objects are those that phased through the thin floor collider at the start/during the capture. Blue objects belong to the avatar; spikes indicate teleportation.

Figure 11 shows prediction error for all sessions. Prediction error is the distance between the x, y, z coordinates of a constraint after evaluation at time t, and the position of the remote object pertaining to that constraint at t. Prediction error is almost identical to position divergence. This is expected, as we do not damp predictions and the only other source of error is local non-determinism, which is very small frame-to-frame.

6.4.2 Latency

As both nodes used the same time-reference, we can trivially measure the effective latency of each update. The effective latency is the time between update generation and evaluation – when the update is extrapolated and applied to the local simulation.

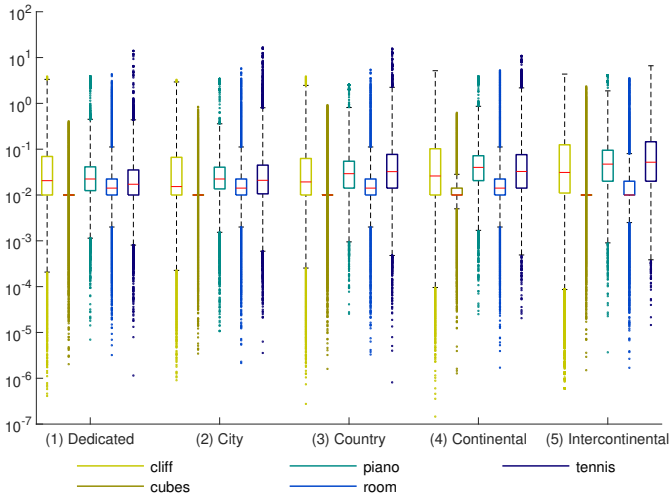

Fig. 11. Prediction error in meters between two nodes for each session.

Task	Mean (ms)	Std. (ms)
Parse Message	0.011	0.008
Process Message	0.019	0.682
Constraint Solve	0.035	0.017

TABLE 4
Overhead per shared-object.

Figure 12 shows the latency distributions of all updates, across all scenarios and profiles. Profiles are scenario independent but the Cliff scenario had a reduced update rate of 10 Hz, and higher computational demands often resulting in frame rate drops that are represented in its inflated latencies. As can be seen the latencies are in line with the expected network performance (Table 3).

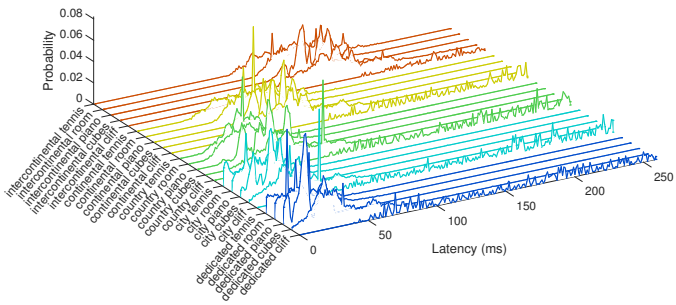

Fig. 12. Probability distributions of effective latencies by session up to 250 ms (grouped by connection type).

6.4.3 Overhead

Overheads for shared-objects are shown in Table 4. Generally these are trivial, however the heap allocations made by the linear-algebra and serialisation libraries caused unpredictable performance dips when garbage-collection was performed. This could be resolved by using a different rotation averaging method or a bespoke implementation, and using a different serialisation library.

While the overhead was insignificant for most scenarios, Cliff, with its 1000 objects, suffered serious slowdown. The update rate had to be decreased to 10 Hz and rotation averaging disabled to run in real-time.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3052580, IEEE Transactions on Visualization and Computer Graphics

6.4.4 Dynamics

We are not aware of a way to objectively measure the aesthetic dynamics of simulated objects to see what impact the network has on their behaviour. As we target non-deterministic simulations, and use real input, there is no ground truth to compare with. To approximate dynamics distortion, we plot the acceleration distributions from each session (Figure 13). This is a very noisy measure. However we can see a skewing of the distribution in line with our observations for some scenarios, such as the Cliff and Tennis.

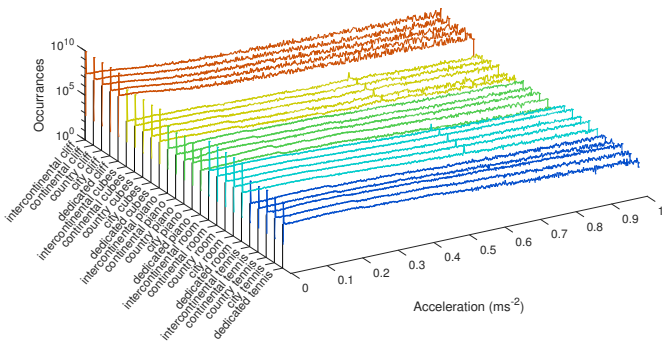

Fig. 13. Acceleration distributions for each session (grouped by scenario type).

7 Discussion

7.1 Modes of Operation

The system can operate in three modes depending on process flow. In the first, all constraints are duplicated and users affect the simulation through shared mechanisms (*soft*). We imagine this to be the most common. In the second, (*hard*) updates are no longer functions of the current state. The system has no simulation abilities, but is completely stable and responsive. In the third, constraints are asymmetric. Remote constraints manifest only through corrections. The nodes behave as one large numerical solver. QoS must be very high to avoid instability.

7.2 Performance

Consistency requirements are application dependent, but our metrics show most scenarios maintain millimeter scale consistency even in the worst-case for the vast majority of objects. Our network profiles deliberately underestimated QoS to demonstrate when and how performance degrades. We presented a dynamics measure (Figure 13) but this does not sufficiently capture performance degredation. The Tennis scenario became unplayable beyond Profile 3 (28.5 ms latency). This was due to the low update-rate of the rackets, and prediction error of the ball. These combined prevented the ball or rackets following trajectories smoothly enough to reliably support interception. Improvements could be to decrease the physics timestep and enable interpolation for intervening rendering frames. Reducing influence would also reduce the effect of prediction error, but at a risk of increasing the probablity of missing a remote collision.

7.3 Physical Plausability

CBN was conceived to be physically plausible rather than physically accurate. Our metrics suggest that this is achieved, but CBN may not be suitable for applications that require a ground truth, which has no concept under continous authority. While CBN has parallels with virtual couplings, it does not transmit power variables so is not physically accurate.

7.4 Correction

The distinguishing characteristic of the CBN solver is that 'correction' is not an action that has a real-world physical analogue. The simulation state is simply changed. The energy in the system may change after correction, but not as a result of the correction mechanism.

7.5 Prediction and Damping

Accurate predictors enable not just consistency, but responsiveness, as error manifests as damping or otherwise incoherent forces. CBN is most sensitive to prediction error when viewing the actions of remote users. High remote authority prevents the local simulation from damping predictor error and can result in spatial jitter at higher latencies. More accurate predictors would help. When making corrections, the results of Savery & Graham's correction evaluation [73] remain directly applicable.

7.6 Stability

In teleoperation, a system is stable if all nodes and communication links are passive. We cannot make this claim for CBN. To our knowledge only static-analysis has proven stability, and static-analysis cannot be applied to discrete channels.

Still, CBN did not destabilise in any of our test conditions, and we did not need to tune any parameters to achieve this. That is not to say we did not observe instability during development, especially with asymmetrical constraints (Section 3.5). Large impulses from irreconcilable constraints is a known issue in force-based simulations and one of the motivations for PBD.

7.7 Settling Time

When the physics simulation starts there is a period of settling during which objects fall or depenetrate. Under high latencies this can result in large velocities being reflected due to prediction error and its corresponding collision response. In all our scenarios these initial jumps damped themselves out without issue, but could result in objects settling far from their initial position or tunneling through thin colliders. The one scenario for which this does not occur is Cliff: because the objects are fixed in place until 'released' by a key press. Something simliar could be implemented that allows the simulations to settle in sync without this oscillation. For example, the authority could be slowly increased over time at the start from zero, or initial corrections could be damped when nodes first communicate.

7.8 Jitter

The biggest weakeness of our evaluation is that we do not measure jitter. Jitter can be more salient than latency [59]. However we are not aware of an objective measure jitter as it relates to user experience. Our closest approximation is the measured dynamics (Section 6.4.4). Anecdotally we did observe the effects of jitter in our evaluation, and identified QoS thresholds at which scenarios became unusable (with the implication that jitter was not significant in better QoS profiles).

7.9 Protocol

Our current implementation uses TCP, which is a reliable protocol. Our implementation does not detect out of order packets but this should be trivial to implement as updates are already timestamped to a shared reference. In principle, our technique should be robust to missing packets as with out-of-order filtering these will manifest as lower update rates.

7.10 Implementation

We consider ease-of-implementation to be a characteristic of CBN. Synchronisation is independent of simulation technique – as evidenced by our demonstrations using PhysX, which is closed-source in Unity. While advanced non-linear constraints and highly accurate predictors can be used for enhanced performance, we have demonstrated acceptable performance using L1-norm & EVD-based averaging and second-order Euler predictors. Only EVD for rotation averaging required a specialised library.

Transactional synchronisation would require graph-colouring and distributed mutex structures. Force-reflection itself is simple, but over a latent link requires the implementation of stabilisation schemes, which increase in complexity with transparency.

In theory CBN is loosely coupled to any engine, but in practice requires an API to change the simulation without introducing energy. An engine with no counterpart to PhysX's *MovePosition*/*MoveRotation* would not be supported. Even PhysX's API's are not ideal, as they do not support continous collision detection for when objects are fully remotely controlled, as was noticable in the Tennis scenario. Additionally we had to manage the PhysX's clock, something that may not be possible on all platforms.

When integrating with a force-based engine, operation ordering is important. For example, the velocity term in PhysX is a function of gravity. So if PhysX is stepped after the CBN solve, the velocity could be non-zero when the state is transmitted to remote nodes or interpolated for the renderer, even if the solved steady-state velocity is zero. Similarly, on platforms such as Unity where the physics engine is a module with its own scene graph (even if inaccessible) the time at which transforms are read is important. The renderer should display the CBN solution to avoid jitter, for example, and it should be the starting point for subsequent physics step(s).

7.11 Applications

Our implementation is a proof-of-concept and there are cases that would need to be handled in a real application, for example where constraints diverge irrecoverably. Our technique also has no mechanism for meta-changes or events. We expect this would be supported using a second channel. CBN does not preclude the use of additional methods, for example, combining with gameplay-determinism.

7.12 Update Rate and Influence

Our fixed update rate is not unrealistic. We set out to support non-deterministic or otherwise heterogenous simulations. Therefore a constant heartbeat is needed to correct for non-determinism, even if only due to, e.g. quantisation noise. Still by modelling expected divergence this rate could be adapted per object to target a specific consistency, like a PCAM. Influence values were hand-calibrated in our experiments. We did not find the simulations to be very sensitive to influence below 10%. Above 20% they became quite sensitive to prediction error.

7.13 Large Scale Simulations

Our primary objective was to support L3 collaboration, however the decoupled nature of distributed data-fusion has additional advantages. While we did not demonstrate it, there is no theoretical reason why different engines could not be synchronised with CBN. The implications would be increased noise per update, due to inherent simulation differences. Hypothetically, we could also couple different resolutions. For example, a low-end user may see a simple cloth simulation, while a desktop user sees the highest quality available. The low-power user could even use the desktop user's simulation for upsampling. It has been suggested that ultra-scale shared environments will not flourish under control of a single organisation [79]. An approach like CBN may be better able to support synchronisation between simulations from multiple vendors. In addition, the focus on autonomous simulations could potentially give users more control over their personal experience of the shared space. A downside with distributed systems however is that they are vulnerable to a 51% attack (a single entity controlling a majority of nodes and so directing the distributed average). Further, these decentralised advantages are tempered by the necessity of an agreement mechansim to correct implausible divergences.

8 Future Works

8.1 Influence

Influence is meant to describe to each simulation the information of an update. In the future we hope this can become a theoretically correct measure of information. A first step could be to model the error of the predictors as a function of state and link properties.

8.2 Bandwidth Optimisation

Nodes communicated in a peer-to-peer fashion. The synchronisation technique is decoupled from the architecture, but using the current architecture will limit scalability. Future work could examine how different network architectures affect convergence, based on the existing work on scene graph synchronisation and consensus algorithms.

8.3 Distributed Position Based Dynamics

Our implementation used PhysX to demonstrate its applicability to current technologies. PBD is in theory more ammeable to CBN as all constraints can be exchanged as energy gradients and there is no division between the networked and local components in the single-step solve. If the network conditions supported it, this could allow for easy building of large simulations based on the asymmetric constraint model, with constraints assigned to compute nodes based on, e.g. spatial coherence. In this way distributed PBD is more similar to constraint based consensus algorithms for general optimisation [48].

9 Conclusion

Supporting concurrent manipulation (L3) is challenging for DVEs because the goals of consistency and responsiveness become contradictory with latency. There are different ways to build DVEs, but each has its own drawbacks. DVEs have been considered as distributed transactional databases, communicating changes in discrete updates. These methods have difficulty supporting L3. Alternatively, force-reflection methods exchange asymmetric

variables that are integrated at a server. This requires stabilisation, and the client-server architecture limits scalability.

With CBN we suggest that DVE synchronisation be considered a distributed data-fusion problem. Autonomous simulations run in parallel, and use distributed averaging to maintain consistency over time. Constraint duplication limits divergence, while local prediction limits damping due to latency. Operating at the position level, CBN, like PBD, is inherently stable. The continous authority used in the distributed average allows L3 collaboration and abstraction of the simulation technique. CBN can couple heterogenous simulations, supporting not just non-deterministic, but completely different simulations, such as those using different engines, or different input modalities, such as haptics.

To validate CBN, we implemented a number of scenarios representative of common challenges in DVEs. These demonstrated support for large scale simulations, causality preservation, complex interactions such as stacking, room-scale L3 collaboration and transparent support for haptics. We evaluated the performance of CBN under different QoS profiles using objective measures of consistency. Our work is early and there are limitations to our implementation and evaluation. We were unable to prevent tunnelling in the scenario designed to demonstrate causality preservation, and we do not have an objective evaluation of jitter. We have had strong successes however. Under typical link properties (up to 60 ms), high consistencies (mm scale) were maintained in many scenarios. We showed support for L3 collaboration without force reflection, including handover and collaborative lifting of multiple stacked objects in a peer-to-peer simulation. In one of the most surprising results our system transparently coupled two haptic devices and maintained stability while treating the simulation as a large numerical solver across a link with over 200 ms of latency.

We believe that considering DVEs as a data-fusion problem, rather than a distributed database or teleoperation problem, can have broad and beneficial implications for large scale simulations. By supporting heterogenous nodes, we can couple different input devices, and perhaps different simulations, of different capabilities. This could have an impact on how multi-vendor shared VEs are administered, and how users could control their personal experience of such spaces. In the future we hope that CBN can be taken further. Specifically, to derive an information-theoretically correct measure of influence, and to improve our evaluation by investigating jitter and networks of larger scale.

ACKNOWLEDGMENTS

The authors wish to thank FSMLabs, NDFIS and JISC. This work was supported by EPSRC grants EP/P004040/1 and EP/P004016/1.

REFERENCES

[1] J. Allard, V. Gouranton, L. Lecointre, S. Limet, E. Melin, B. Raffin, and S. Robert. FlowVR: A Middleware for Large Scale Virtual Reality Applications. In *Parallel Processing. 10th International Euro-Par Conference*, pages 497–505. Springer Berlin Heidelberg, 2004.

[2] R. S. Allison, J. E. Zacher, D. Wang, and J. Shu. Effects of network delay on a collaborative motor task with telehaptic and televisual feedback. *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 375–381, 2004.

[3] C. Anthes, R. Landertshamer, H. Bressler, and J. Volkert. Managing Transformations and Events in Networked Virtual Environments. In *Advances in Multimedia Modeling. MMM 2007. Lecture Notes in Computer Science, vol 4352*, pages 722–729, 2006.

[4] H. Arioui, A. Kheddar, and S. Mammar. A predictive wave-based approach for time delayed virtual environments haptics systems. In *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*, pages 134–139. IEEE, 2002.

[5] P. Berestesky, N. Chopra, and M. Spong. Discrete time passivity in bilateral teleoperation over the Internet. In *Proceedings. IEEE International Conference on Robotics and Automation 2004. ICRA '04.*, volume 5, pages 4557–4564. IEEE, 2004.

[6] Y. W. Bernier. Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization. Technical report, Valve, 2001.

[7] A. Boukerche, S. Shirmohammadi, and A. Hossain. Prediction-based decorators for distributed collaborative haptic virtual environments. *International Journal of Computer Applications in Technology*, 29(1):81, 2007.

[8] F. Brandi and E. Steinbach. Prediction techniques for haptic communication and their vulnerability to packet losses. In *2013 IEEE International Symposium on Haptic Audio Visual Environments and Games (HAVE)*, pages 63–68. IEEE, 10 2013.

[9] W. Broll. Interacting in distributed collaborative virtual environments. In *Proceedings Virtual Reality Annual International Symposium '95*, pages 148–155. IEEE Comput. Soc. Press, 1995.

[10] P. Buttolo, R. Oboe, and B. Hannaford. Architectures for shared haptic virtual environments. *Computers & Graphics*, 21(4):421–429, 1997.

[11] F. Chardavoine, S. Ageneau, and B. Ozell. Wolverine: A Distributed Scene-Graph Library. *Presence: Teleoperators and Virtual Environments*, 14(1):20–30, 2005.

[12] A. Chatterjee and V. M. Govindu. Efficient and robust large-scale rotation averaging. *Proceedings of the IEEE International Conference on Computer Vision*, pages 521–528, 2013.

[13] D. Chitimalla, K. Kondepu, L. Valcarenghi, M. Tornatore, and B. Mukherjee. 5G FronthaulLatency and Jitter Studies of CPRI Over Ethernet. *Journal of Optical Communications and Networking*, 9(2):172, 2 2017.

[14] Z. Choukair, D. Retailleau, and M. Hellstrom. Environment for performing collaborative distributed virtual environments with QoS. In *Proceedings of the Seventh International Conference on Parallel and Distributed Systems*, pages 111–118. IEEE Comput. Soc, 2000.

[15] E. Coelho, B. MacIntyre, and S. Julier. OSGAR: A Scene Graph with Uncertain Transformations. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 6–15. IEEE, 2004.

[16] E. Cronin, A. R. Kurc, B. Filstrup, and S. Jamin. An efficient synchronization mechanism for mirrored game architectures. *Multimedia Tools and Applications*, 23(1):7–30, 2004.

[17] D. Delaney, T. Ward, and S. McLoone. On Consistency and Network Latency in Distributed Interactive Applications: A Survey - Part II. *Presence*, 15(4):465–482, 2006.

[18] D. Delaney, T. Ward, and S. McLoone. On Consistency and Network Latency in Distributed Interactive Applications: A SurveyPart I. *Presence: Teleoperators and Virtual Environments*, 15(2):218–234, 4 2006.

[19] F. Drolet, M. Mokhtari, F. Bernier, and D. Laurendeau. A Software Architecture for Sharing Distributed Virtual Worlds. In *Proceedings of the 2009 IEEE Virtual Reality Conference*, pages 271–272. IEEE, 3 2009.

[20] Exit Games. PUN Introduction. https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro, 2020. [Online; accessed 25-Feb-2020].

[21] G. Fiedler. https://developer.oculus.com/blog/networked-physics-in-virtual-reality-networking-a-stack-of-cubes-with-unity-and-physx/, 2018. [Online; accessed 30-Aug-2019].

[22] R. Fujimoto. Parallel and distributed simulation systems. In *Proceeding of the 2001 Winter Simulation Conference (Cat. No.01CH37304)*, pages 147–157. IEEE, 2001.

[23] F. Garin and L. Schenato. Distributed estimation and control applications using linear consensus algorithms. *Networked Control Systems*, page 75107, 2011.

[24] S. Ghapani, S. Rahili, and W. Ren. Distributed Average Tracking of Physical Second-Order Agents with Heterogeneous Unknown Nonlinear Dynamics Without Constraint on Input Signals. *IEEE Transactions on Automatic Control*, 64(3):1178–1184, 2019.

[25] C. Greenhalgh and S. Benford. MASSIVE. *ACM Transactions on Computer-Human Interaction*, 2(3):239–261, 9 1995.

[26] I. J. Grimstead, D. W. Walker, and N. J. Avis. Collaborative visualization: A review and taxonomy. *Proceedings - IEEE International Symposium on Distributed Simulation and Real-Time Applications, DS-RT*, pages 61–69, 2005.

[27] C. Gunn, M. Hutchins, and M. Adcock. Combating Latency in Haptic Collaborative Virtual. *Presence*, 14(3):313–328, 2005.

[28] C. Gunn, M. Hutchins, D. Stevenson, M. Adcock, and P. Youngblood. Using Collaborative Haptics in Remote Surgical Training. In *First Joint*

Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, pages 481–482. IEEE, 2005.

[29] C. Gutwin, J. Dyck, and J. Burkitt. Using cursor prediction to smooth telepointer jitter. In *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work - GROUP '03*, page 294, New York, New York, USA, 2003. ACM Press.

[30] D. Hanawa and T. Yonekura. On the error modeling of dead reckoned data in a distributed virtual environment. *Advanced Modeling and Optimization*, 7(1):85–98, 2005.

[31] D. Hanawa and T. Yonekura. A Proposal of Dead Reckoning Protocol in Distributed Virtual Environment based on the Taylor Expansion. In *2006 International Conference on Cyberworlds*, pages 107–114. IEEE, 2006.

[32] R. Hartley, J. Trumpf, Y. Dai, and H. Li. Rotation averaging. *International Journal of Computer Vision*, 103(3):267–305, 2013.

[33] T. Hatanaka, N. Chopra, M. Fujita, and M. W. Spong. *Passivity-Based Control and Estimation in Networked Robotics*. 2015.

[34] G. Hesina, D. Schmalstieg, A. Furhmann, and W. Purgathofer. Distributed Open Inventor. In *Proceedings of the ACM symposium on Virtual reality software and technology - VRST '99*, pages 74–81, New York, New York, USA, 1999. ACM Press.

[35] P. Hinterseer, E. Steinbach, and S. Chaudhuri. Perception-Based Compression of Haptic Data Streams Using Kalman Filters. In *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings*, volume 5. IEEE, 2006.

[36] P. F. Hokayem and M. W. Spong. Bilateral teleoperation: An historical survey. *Automatica*, 42(12):2035–2057, 2006.

[37] R. Hoskinson. Determinism ni League Of Legends: Implementation. Technical report, Riot Games, 2017.

[38] Joono Cheong, S. Niculescu, A. Annaswamy, and M. Srinivasan. Motion Synchronization in Virtual Environments with Shared Haptics and Large Time Delays. In *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 277–282. IEEE, 2005.

[39] P. Jorissen, M. Wijnants, and W. Lamotte. Dynamic Interactions in Physically Realistic Collaborative Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics*, 11(6):649–660, 11 2005.

[40] J. Y. Jung, B. D. Adelstein, and S. R. Ellis. Discriminability of Prediction Artifacts in a Time-Delayed Virtual Environment. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 44(5):499–502, 7 2000.

[41] D. Lake, M. Bowman, and H. Liu. Distributed scene graph to enable thousands of interacting users in a virtual environment. In *9th Annual Workshop on Network and Systems Support for Games*, pages 1–6. IEEE, 11 2010.

[42] B. M. Lambeth, J. LaPlant, E. Clapan, and F. G. Hamza-Lup. The effects of network delay on task performance in a visual-haptic collaborative environment. In *Proceedings of the 47th Annual Southeast Regional Conference - ACM-SE 47*, New York, New York, USA, 2009. ACM Press.

[43] M. E. Latoschik, C. Fröhlich, and A. Wendler. Scene Synchronization in Close Coupled World Representations using SCIVE. *The International Journal of Virtual Reality*, 5(3):47–52, 2006.

[44] D. A. Lawrence. Stability and Transparency in Bilateral Teleoperation. *IEEE Transactions on Robotics and Automation*, 9(5):624–637, 1993.

[45] R. Lozano, N. Chopra, and M. W. Spong. Passivation of force reflecting bilateral teleoperators with time varying delay. In *Proceedings of the 8th Mechatronics Forum*, pages 954–962, 2002.

[46] M. Lysenko. Replication in networked games. *0 FPS*, 2014.

[47] B. MacIntyre and S. K. Feiner. A distributed 3D graphics library. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, pages 361–370, 1998.

[48] K. Margellos, A. Falsone, S. Garatti, and M. Prandini. Distributed Constrained Optimization and Consensus in Uncertain Networks via Proximal Minimization. *IEEE Transactions on Automatic Control*, 63(5):1372–1387, 2018.

[49] D. Margery, B. Arnaldi, and N. Plouzeau. A General Framework for Cooperative Manipulation in Virtual Environments. In *Proceedings of the Eurographics Workshop in Vienna, Austria, May 31-June 1, 1999*, Eurographics, pages 169–178, Vienna, 1999. Springer Vienna.

[50] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 4 2007.

[51] M. Naef, E. Lamboray, O. Staadt, and M. Gross. The blue-c distributed scene graph. In *Proceedings of the 2003 IEEE Virtual Reality Conference*, pages 275–276. IEEE Comput. Soc, 2003.

[52] G. Niemeyer and J.-J. Slotine. Stable adaptive teleoperation. *IEEE Journal of Oceanic Engineering*, 16(1):152–162, 1 1991.

[53] G. Niemeyer and J. J. E. Slotine. Telemanipulation with time delays. *International Journal of Robotics Research*, 23(9):873–890, 2004.

[54] NTT. IP Network Service Level Agreement Performance Statistics. https://www.ntt.com/en/services/network/gin/sla/stats.html, 2019. [Online; accessed 10-Aug-2019].

[55] E. Nuño, R. Ortega, L. Basañez, and D. Hill. Synchronization of networks of nonidentical euler-lagrange systems with uncertain parameters and communication delays. *IEEE Transactions on Automatic Control*, 56(4):935–941, 2011.

[56] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, 95(1):215–233, 1 2007.

[57] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[58] L. Pantel and L. C. Wolf. On the suitability of dead reckoning schemes for games. In *Proceedings of the 1st workshop on Network and system support for games - NETGAMES '02*, pages 79–84, New York, New York, USA, 2002. ACM Press.

[59] K. Park and R. Kenyon. Effects of network characteristics on human performance in a collaborative virtual environment. *Proceedings of the 1999 IEEE Virtual Reality Conference*, 1999.

[60] R. Pusch. Explaining how fighting games use delay-based and rollback netcode. *Ars Technica*, 2019.

[61] J. Qin, K.-S. Choi, R. Xu, W.-M. Pang, and P.-A. Heng. Effect of Packet Loss on Collaborative Haptic Interactions in Networked Virtual Environments: An Experimental Study. *Presence: Teleoperators and Virtual Environments*, 22(1):36–53, 2 2013.

[62] D. Roberts and P. Sharkey. Minimising the latency induced by consistency control, within a large scale multi-user distributed virtual reality system. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 5, pages 4492–4497. IEEE, 1997.

[63] D. Roberts and R. Wolff. Controlling Consistency within Collaborative Virtual Environments. In *Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 46–52. IEEE, 2004.

[64] E. RodrIguez-Seda, Dongjun Lee, and M. Spong. Experimental Comparison Study of Control Architectures for Bilateral Teleoperators. *IEEE Transactions on Robotics*, 25(6):1304–1318, 12 2009.

[65] M. Roth, G. Voss, and D. Reiners. Multi-threading and clustering for scene graph systems. *Computers and Graphics (Pergamon)*, 28(1):63–66, 2004.

[66] R. A. Ruddle, J. C. D. Savage, and D. M. Jones. Symmetric and asymmetric action integration during cooperative object manipulation in virtual environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 9(4):285–308, 2002.

[67] M. Ryan and P. Sharkey. Causal volumes in distributed virtual reality. In *IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, pages 1067–1072. IEEE, 1997.

[68] M. Ryan and P. Sharkey. The causal surface and its effect on distribution transparency in a distributed virtual environment. In *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, volume 6, pages 75–80. IEEE, 1999.

[69] M. D. Ryan and P. M. Sharkey. Distortion in Distributed Virtual Environments. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1434, pages 42–48. 1998.

[70] P. Sandoz, P. Sharkey, and D. Roberts. Collision prediction of a moving object within a virtual world. *VR World*, 96(March):1315, 1996.

[71] G. Sankaranarayanan and B. Hannaford. Virtual Coupling Schemes for Position Coherency in Networked Haptic Environments. In *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 853–858. IEEE, 2006.

[72] A. Sarlette, R. Sepulchre, and N. E. Leonard. Autonomous rigid body attitude synchronization. In *46th IEEE Conference on Decision and Control*, pages 2566–2571, 2007.

[73] C. Savery and N. Graham. Reducing the negative effects of inconsistencies in networked games. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play - CHI PLAY '14*, pages 237–246, New York, New York, USA, 2014. ACM Press.

[74] A. Schiefer, R. Berndt, T. Ullrich, V. Settgast, and D. W. Fellner. Service-oriented scene graph manipulation. *Web3D Symposium Proceedings*, 1(212):55–62, 2010.

[75] C. Schuwerk, R. Chaudhari, and E. Steinbach. Delay compensation in Shared Haptic Virtual Environments. In *Proceedings of the 2014 IEEE Haptics Symposium (HAPTICS)*, pages 371–377. IEEE, 2 2014.

[76] P. Sharkey, M. Ryan, and D. Roberts. A local perception filter for distributed virtual environments. In *Proceedings. IEEE 1998 Virtual Reality Annual International Symposium*, pages 242–249. IEEE Comput. Soc, 1998.

[77] S. K. Singhal and D. R. Cheriton. Exploiting Position History for Efficient Remote Rendering in Networked Virtual Reality. *Presence: Teleoperators and Virtual Environments*, 4:169–193, 1995.

[78] A. Steed and M. F. Oliveira. *Networked Graphics*. Morgan Kaufmann, 2010.

[79] T. Sweeney. Foundational principles & technologies for the metaverse. In *ACM SIGGRAPH 2019 Talks on - SIGGRAPH '19*, pages 1–1, New York, New York, USA, 2019. ACM Press.

[80] M. Tavakoli, A. Aziminejad, R. V. Patel, and M. Moallem. Methods and mechanisms for contact feedback in a robot-assisted minimally invasive environment. *Surgical Endoscopy*, 20(10):1570–1579, 10 2006.

[81] M. Terrano and P. Bettner. 1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond. *Gamasutra*, 2001.

[82] A. Tumanov, R. Allison, and W. Stuerzlinger. Variability-Aware Latency Amelioration in Distributed Environments. In *Proceedings of the 2007 IEEE Virtual Reality Conference*, pages 123–130. IEEE, 2007.

[83] A. Valadares, T. Debeauvais, and C. V. Lopes. Evolution of scalability with synchronized state in virtual environments. In *2012 IEEE International Workshop on Haptic Audio Visual Environments and Games (HAVE 2012) Proceedings*, pages 142–147. IEEE, 10 2012.

[84] Verizon. IP Latency Statistics. https://enterprise.verizon.com/terms/latency/, 2019. [Online; accessed 10-Aug-2019].

[85] G. Voß, J. Behr, D. Reiners, and M. Roth. A multi-thread safe foundation for scene graphs and its extension to clusters. In *EGPGV '02: Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, pages 33–37, 2002.

[86] G. Wang, S. Li, S. Wang, B. Lu, and W. Li. ViWoSG: A distributed scene graph of ultramassive distributed virtual environments. *Science in China, Series F: Information Sciences*, 52(3):457–469, 2009.

[87] S. T. Watt, S. Achanta, H. Abubakari, and E. Sagen. Understanding and Applying Precision Time Protocol. In *Power and Energy Automation*, number March 2014, 2015.

[88] R. Wolff, D. J. Roberts, and O. Otto. A Study of Event Traffic During the Shared Manipulation of Objects Within a Collaborative Virtual Environment. *Presence: Teleoperators and Virtual Environments*, 13(3):251–262, 6 2004.

[89] L. Xiao, S. Boyd, and S.-J. Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, 1 2007.

[90] Ya Zhang and Yu-Ping Tian. Consensus of Data-Sampled Multi-Agent Systems With Random Communication Delay and Packet Loss. *IEEE Transactions on Automatic Control*, 55(4):939–943, 4 2010.

[91] M. Zaeh, S. Clarke, P. Hinterseer, and E. Steinbach. Telepresence Across Networks: A Combined Deadband and Prediction Approach. In *Tenth International Conference on Information Visualisation (IV'06)*, pages 597–604. IEEE, 2006.

[92] B. Zeleznik, L. Holden, M. Capps, H. Abrams, and T. Miller. Scene-Graph-As-Bus: Collaboration between Heterogeneous Stand-alone 3-D Graphical Applications. *Computer Graphics Forum*, 19(3):91–98, 2000.